Why the Best Predictive Models Are Often Different from the Best Explanatory Models: A Theoretical Explanation

Songsak Sriboonchitta, Luc Longpré, Vladik Kreinovich, and
Thongchai Dumrongpokaphan

Abstract Traditionally, in statistics, it was implicitly assumed that models which are the best predictors also have the best explanatory power. Lately, many examples have been provided that show that the best predictive models are often different from the best explanatory models. In this paper, we provide a theoretical explanation for this difference.

1 Formulation of the Problem

Predictive models vs. explanatory models: a traditional confusion. Traditionally, many researchers who have applied statistical methods implicitly assumed that predictive and explanatory powers are strongly correlated:

- they assumed that a statistical model that leads to accurate predictions also provides a good explanation for the corresponding phenomenon, and
- they also assumed that models providing a good explanation for the observed phenomena also lead to accurate predictions.

Predictive models vs. explanatory models: a general distinction. In practice, models that lead to good predictions do not always explain the observed phenom-

Songsak Sriboonchitta
Faculty of Economics, Chiang Mai University, Thailand
e-mail: songsakecon@gmail.com

Luc Longpré and Vladik Kreinovich
University of Texas at El Paso, El Paso, Texas 79968, USA
e-mail: longpre@utep.edu, vladik@utep.edu

Thongchai Dumrongpokaphan
Department of Mathematics, Faculty of Science, Chiang Mai University, Thailand
e-mail: tcd43@hotmail.com

ena. Vice versa, models that nicely explain the corresponding phenomena do not always lead to most accurate predictions.

To illustrate the difference, let us give a simple example from celestial mechanics; see, e.g., [1]. Newton's equations provide a very clear explanation of why and how celestial bodies move, why the planets and satellites follow their orbits, etc. In principle, we can predict the trajectories of celestial bodies – and thus, their future observed positions in the sky – by directly integrating the corresponding differential equations. This would, however, require a lot of computation time on modern computers.

On the other hand, people successfully predicted the observed positions of planets way before Newton: for that, they use *epicycles*, i.e., in effect, trigonometric series. Such series are still used in celestial mechanics to predict the positions of celestial bodies. They are very good for predictions, but they are absolutely useless in explanations.

Predictive models vs. explanatory models in statistics. In statistics, the need to differentiate between predictive and explanatory models was emphasized and illustrated by Galit Shmueli in [3].

Remaining problem: why? The empirical fact that the best predictive models are often different from the best explanatory models is currently well known and well recognized.

But from the theoretical viewpoint, this empirical fact still remains a puzzle. In this paper, we provide a theoretical explanation for this empirical phenomenon.

2 Towards Formal (Precise) Definitions: Analysis of the Problem

Need for formalization. In order to provide a theoretical explanation for the difference between the best predictive and the best explanatory models, we need to first formally describe:

- what it means for a model to be the best predictive model, and
- what it means for a model to be the best explanatory model.

What does it mean for a model to be explanatory: analysis of the problem. The "explanatory" part is intuitively understandable: we have some equations or formulas that *explain* all the observed data – in the sense that all the observed data satisfy these equations.

Of course, these equations must be checkable – otherwise, if they are formulated purely in terms of complex abstract mathematics, so that no one knows how to check whether observed data satisfy these equations or formulas, then how can we know that the data satisfies them?

Thus, when we say that we have an explanatory model, what we are saying, in effect, that we have an algorithm – a program if you will – that, given the data, checks

whether the data is consistent with the corresponding equations or formulas. From this pragmatic viewpoint, by an explanatory model, we simply means a program.

Of course, this program must be non-trivial: it is not enough for the data to be simply *consistent* with the data, explanatory means that we must *explain* all this data. For example, if we simply state that, in general, the trade volume grows when the GDP grows, all the data may be consistent with this rule, but this consistency is not enough: for a model to be truly explanatory, it needs to explain *why* in some cases, the growth in trade is small and in other cases, it is huge. In other words, it must explain the exact growth rate. Of course, this is economics, not fundamental physics, we cannot explain all the numbers based on first principles only, we have to take into account some quantities that affect our processes. But for the model to be truly explanatory we must be sure that, once the values of these additional quantities are fixed, there should be only one sequence of numbers that satisfies the corresponding equations or formulas – namely, the sequence that we observe (ignoring noise, of course).

This is not that different from physics. For example, Newton's laws of gravitation allow many possible orbits of celestial bodies, but once you fix the masses, initial conditions, and initial velocities of all these bodies, then Newton's laws uniquely determine how these bodies will move.

In algorithmic terms, if:

- to the original program for checking whether the data satisfies the given equations and/or formulas,
- we add auxiliary parts checking whether the values of additional quantities are exactly the ones needed to explain the data,
- then the observed data is the only possible sequence of observations that is consistent with this program.

Once we know such a program that uniquely determines all the data, we can, in principle, find this data – i.e., solve the corresponding equations – by simply trying all possible combinations of possible data values until we find the one that satisfies all the corresponding conditions.

How can we describe this in precise terms? All the observations can be stored in the computer, and in the computer, everything is stored as 0s and 1s. From this viewpoint, the whole set of observed data is simply a binary sequence x, i.e., a finite sequence of 0s and 1s.

The length n of this sequence is known. We know how many binary sequences there are of each length:

- there are 2 sequences of length 1: 0 and 1;
- there are $2 \times 2 = 2^2 = 4$ sequences of length 2:
  - two sequences 00 and 01 that start with 0 and
  - two sequences 10 and 11 that start with 1;
- there are $2^2 \times 2 = 2^3 = 8$ binary sequences of length 3:
  - two sequences 000 and 001 that start with 00,

  - two sequences 010 and 011 that start with 01,
  - two sequences 100 and 101 that start with 10, and
  - two sequences 110 and 111 that start with 11;
- in general, we have 2^n sequences of length n.

There are finitely many such sequences, so we must potentially check them all and thus, find the desired sequence x – the only one that satisfies all the required conditions.

Of course, for large n, the time 2^n can be unrealistically astronomically large, so we are talking about *potential* possibility to compute – not practical computations: one does not solve Newton's equations by trying all possible trajectories and checking whether they satisfy Newton's equations. But it is OK, since our goal here is not to provide a practical solution to the problem, but rather to provide a formal definition of an explanatory model.

For the purpose of this definition, we can associate each explanatory model not only with the original checking program, but also with the related exhaustive-search program p that generates the data. The exhaustive search part is easy to program, it practically does not add to length of the original checking program. So, we arrive at the following definition.

Definition 1. *Let a binary sequence x be given. We will call this sequence* data. *By an* explanatory model*, we mean a program p that generates the binary sequence x.*

Comments.

- The above definition, if we read it without the previous motivations part, sounds very counter-intuitive. However, we hope that the motivation part has convinced the reader that this strange-sounding definition indeed describes what we usually mean by an explanatory model.
- For each data, there is at least one explanatory model – since we can always have a program that simply prints all the bits of the given sequence x one by one.

What do we mean by the best explanatory model: analysis of the problem. There are usually several possible explanatory models, which of them is the best?

To formalize this intuitive notion, let us again go back to physics. Before Newton, the motion of celestial bodies was described by epicycles. To accurately describe the motion of each planet, we needed to know a large number of parameters:

- in the first approximation, in which the orbit is a circle, we need to know the radius of this circle, the planet's initial position on this circle, and its velocity;
- in the second approximation, we need to know similar parameters of the first auxiliary circular motion that describes the deviation from the circle;
- in the third approximation, we need to know similar parameters of the second auxiliary circular motion describing the deviation from the second-approximation trajectory, etc.

Then came Kepler's idea that celestial bodies follow elliptical trajectories. Why was this idea better than epicycles? Because now, to describe the trajectory of each celestial body, we need fewer parameters: all we need is a few parameters that describe the corresponding ellipse.

These original parameters formed the main part of the corresponding data checking program – and thus, of the resulting data generating program. By reducing the number of such parameters, we thus drastically reduced the length of the checking program – and thus, of the generating program corresponding to the model.

Similarly, what Newton did was replaced all the parameters of the ellipses by a few parameters describing the bodies themselves – and this described not only the regular motion of celestial bodies, he also described the tides, he described (explained) why apples from a tree fall down and how exactly, etc. Here, we also have fewer parameters needed to explain the observed data – and thus, a much shorter generating program.

From this viewpoint, a model is better if its generating program is shorter – and thus, the best explanatory model is the one which is the shortest, i.e., the one for which the (bit) length $\mathrm{len}(p)$ of the corresponding program p is the smallest possible. So, we arrive at the following definition.

Definition 2. *Let x be the data. We say that an explanatory model p_0 for x is the* best explanatory model *if it is the shortest of all explanatory models for x, i.e., if*

$$\mathrm{len}(p_0) = \min\{\mathrm{len}(p) : p \text{ generates } x\}.$$

What do we mean by the best predictive model. Clearly, not all models which are explanatory models in the sense of Definition 1 can be used for practical predictions. If using a model requires the astronomical time 2^n of billions of years, then the corresponding program is practically useless:

- if we need thousands of years to predict next year's position of the Moon, we do not need this program: we can as well wait a year and see whether the Moon is;
- similarly, if a trade model takes 10 years of intensive computations to predict next year's trade balance, we do not need this program: we can as well wait a year and see for ourselves.

For a model to be useful for predictions, it needs not just to generate the data x but to generate them *fast* – as fast as possible. The corresponding overall computation time includes both the time needed to upload this program into a computer – which is proportional to the length $\mathrm{len}(p)$ of this program – and the time $t(p)$ needed to run this program.

From this viewpoint, the smaller this overall time $\mathrm{len}(p)+t(p)$, the better. Thus, the best predictive model is the one for which this overall time is the smallest possible. So, we arrive at the following definition.

Definition 3. *Let x be the data. We say that a model p_0 is the* best predictive model for *x if its overall time $\mathrm{len}(p_0)+t(p_0)$ is the smallest among all the explanatory*

models:

$$\text{len}(p_0)+t(p_0)=\min\{\text{len}(p)+t(p) : p \text{ generates } x\}.$$

3 Main Result: Formulation and Discussion

Now that we have formal definitions, we can formulate our main result. It comes as two propositions.

Proposition 1. *No algorithm is possible that, given data x, generates the best explanatory model for this data.*

Proposition 2. *There exists an algorithm that, given data x, generates the best predictive model for this data.*

Discussion. These two results clearly explain why in many cases, the best predictive models are different from the best explanatory models:

- if they were always the same, then the algorithm from Proposition 2 would also always generate the best explanatory models, but
- we know, from Proposition 1, that such a general algorithm is not possible.

4 Proofs

Discussion. It is usually easier to prove that an algorithm exists – all we need to do is to provide such an algorithm. On the other hand, proving that an algorithm does not exist is rarely easy: we need to provide some general arguments why no tricks can lead to such an algorithm.

From this viewpoint, it is easier to prove Proposition 2 than Proposition 1. Let us therefore start with proving Proposition 2.

Proof of Proposition 2. In line with the above idea, let us describe the corresponding algorithm. In this algorithm, to find the program that generates the given data x in the shortest possible overall time T, we start with $T=1$, then take $T=2$, $T=3$, etc. – until we find the smallest value T for which such a program exists.

For each T, we need to look for programs from which $\text{len}(p)+t(p)=T$. For such programs, we have $\text{len}(p)\le T$, so we can simply try all possible binary sequences p of length not exceeding T. There are finitely many strings of each length, so there are finitely many strings p of length $\text{len}(p)\le T$, and we can try try them all.

For each of these strings, we first use a compiler to check whether this string is indeed a syntactically correct program. If it is not, we simply dismiss this string. If the string p is a syntactically correct program, we run it for time $t(p)=T-\text{len}(p)$, to make sure that the overall time is indeed T. If after this time, the program p generates the desired sequence x, this means that we have found the desired best

predictive model, so we can stop – the fact that we did not stop our procedure earlier, when we tested smaller values of the overall time means that no program can generate x in overall time $< T$ and thus, the overall time T is indeed the smallest possible.

The proposition is proven.

Comment. An attentive reader has probably noticed that this algorithm is an exhaustive-search-type algorithm, that requires exponential time 2^n. Yes, this algorithm is not practical – but practicality is not our goal. Our goal is to explain the difference between the best predictive and the best explanatory model, and from the viewpoint of this goal, this slow algorithm serves its purpose: it shows that:

- the best predictive models can be computed by *some* algorithm, while,
- as will now prove, the best explanatory models *cannot* be computed by *any* algorithm – even by a very slow one.

Proof of Proposition 1. The main idea behind this proof comes from the fact that the quantity

$$K(x) \stackrel{\text{def}}{=} \min\{\text{len}(p) : p \text{ generates } x\}$$

is well known in theoretical computer science: it was invented by the famous statistician A. N. Kolmorogov and it is thus known as *Kolmogorov complexity*; see, e.g., [2]. One of the first results that Kolmogorov proved about his new notion is that no algorithm is possible that, given a binary sting x, would always compute its Kolmogorov complexity $K(x)$ [2].

This immediately implies our Proposition 1: indeed, if it was possible to produce, for each data x, the best explanatory model p_0, then we would be able to compute its length $\text{len}(p_0)$ which is exactly $K(x)$ – and $K(x)$ is not computable.

The proposition is proven.

Discussion. It is worth mentioning that the notion of Kolmogorov complexity was originally introduced for a somewhat related but still completely different purpose – how to separate random from non-random sequences.

In the traditional statistics, the very idea that some individual sequences are random and some are not was taboo, one could only talk about probabilities of different sequences. However, intuitively, everyone understands that while a sequence of bits generated by flipping a coin many times is random, a sequence like 010101...01 in which 01 is repeated million times is clearly not random. How can we formally explain this intuitive difference?

Kolmogorov notices that a sequence 0101...01 is not random because it can be generated by a very short program: just repeat 01 many times. For example, in Java, this program looks like this:

```
for(i = 1; i < 1000000; i++)
  {System.out.println("01");}
```

On the other hand, if a sequence is truly random, there is no dependency between different bits, so the only way to print this sequence is to literally print the whole sequence bit by bit:

```
System.out.println("01...");
```

So, when x is not random, we can have short programs generating x. Thus, the shortest possible length $K(x)$ of a program generating x is much smaller than the length $\text{len}(x)$ of this sequence:

$$K(x) \ll \text{len}(x).$$

On the other hand, for a truly random sequence x, you cannot generate it by a program shorter than the above line whose length is $\approx \text{len}(x)$. So, in this case,

$$K(x) \approx \text{len}(x).$$

This idea inspired Kolmogorov to define what we now call Kolmogorov complexity $K(x)$ and to define a binary sequence *random* if $K(x) \geq \text{len}(x) - c_0$, for some appropriate constant c_0.

Proof that Kolmogorov complexity is not computable: reminder. In our proof of Proposition 1 we used Kolmogorov's proof that Kolmogorov complexity $K(x)$ is not computable. To make our result more intuitive, it is worth mentioning that that proof is reasonably intuitive.

The main idea behind this proof comes from the following *Barry's paradox*. Some English expressions describe numbers. For example:

- "twelve" means 12,
- "million" means 1000000, and
- "the smallest prime number larger than 100" means 101.

There are finitely many words in the English language, so there are finitely many combinations of less than twenty words, thus finitely many numbers which can be described by such combinations. Hence, there are numbers which cannot be described by such combinations. Let n_0 denote the smallest of such numbers. Therefore, n_0 is "the smallest number that cannot be describe in fewer than twenty words". But this description of the number n_0 consists of 12 words – less than 20, so n_0 *can* be described by using fewer than twenty words – a clear paradox.

This paradox is caused by the imprecision of natural language, but if we replace "described" by "computed", we get a proof by contradiction that Kolmogorov complexity is not computable.

Indeed, let us assume that $K(x)$ is computable, and let L be the length of the program that computes $K(x)$. Binary sequences can be interpreted as binary integers, so we can talk about the smallest of them. Then, the following program computes the smallest sequence x for which $K(x) \geq 3L$: we try all possible binary sequences of length 1, length 2, etc., until we find the first sequence for which $K(x) \geq 3L$:

```
int x = 0;
while(K(x) < 3 * L){x++;}
```

This program adds just two short lines to the length-L program for computing $K(x)$; thus, its length is $\approx L \ll 3L$, so for the number x_0 that it computes, its Kolmogorov

complexity – the length of the shortest program generating x_0 – cannot exceed this length. Thus, we have $K(x) \ll 3L$.

On the other hand, we defined x_0 as the smallest number for which $K(x) \geq 3L$, so we have $K(x_0) \geq 3L$ – a contradiction. This contradiction shows that our assumption is wrong, and the Kolmogorov complexity is not computable.

Acknowledgments

This work was supported by the Center of Excellence in Econometrics, Faculty of Economics, Chiang Mai University, Thailand. We also acknowledge the partial support of Department of Mathematics, Chiang Mai University, and of the US National Science Foundation via grant HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are greatly thankful to Professors Hung T. Nguyen and Galit Shmueli for valuable discussions.

References

1. R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.
2. M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, Berlin, Heidelberg, New York, 2008.
3. G. Shmueli, "To explain or to predict?", *Statistical Science*, 2010, Vol. 25, No. 3, pp. 289–310.